Eclipse Plugin Tool for Learning Programming Style of Java

Yuki Arakawa*[1], Masayuki Arai[2]

[1]Department of Human Information System, School of Science and Engineering, Teikyo University, Japan

[2] Graduate School of Science and Engineering, Teikyo University, Japan

[2]arai@ics.teikyo-u.ac.jp

Abstract

An Eclipse plugin tool contributive to learning the programming style of Java has been developed in this paper. The tool has the following functions: (1) recommendation on the use of CamelCase and English words for the names of classes, methods, and variables; (2) recommendation on setting the correct scope level of variables and the appropriate length of variable names; (3) recommendation on making comments in source programs; (4) showing sample source programs according to the programming style of Java.

Keywords

Java; Learning Tool; Eclipse plugin; Programming Style; CamelCase; Scope

Introduction

Programming style is very important in the development of programs. Implementing a programming style makes source programs more readable and understandable, and can decrease the number of bugs in a program (D. D. Ward, 2004). Generally, novice programmers treat the programming style lightly. In spite of its importance, most universities and colleges do not teach the programming style to students in programming courses. Consequently, most students do not learn the programming style. To solve this problem, a tool contributive to learning the programming style of Java has been developed(M. Arai, 2012) and implemented as an Eclipse(Eclipse, 2012) plugin. In this paper, the Eclipse plugin tool to learn programming style of Java has been described.

Other learning systems have been proposed. For example, some learning systems detect ill-formed patterns in a program and diagnose the programming style (R. Sekimoto, 2000), while other learning systems detect the bugs or pitfalls of programming(M. Oda, 1994). These systems can detect programming codes that fail to follow the guidelines of the programming style registered in the systems in advance, and therefore users can check for improper programming style. However, the systems can only detect parts of a program, not a whole program.

In contrast, a system supporting module division using information on the active variables in a data flow analysis has been proposed to support a whole program (K. Suzuki, 2004). The system can support comments, names of variables, and whole programs written in the Pascal programming language.

In this paper, a tool has been proposed that has the following functions for novice Java programmers: supports indents and comments, recommends the use of CamelCase and English words for naming, and recommends setting the correct scope level of variables and the appropriate length of variable names. Many systems have the function of supporting indents (Eclipse, 2012) (mule, 2012) and comments (K. Suzuki, 2004). However, the other functions proposed above are new.

Outline of the Tool

Fig. 1 depicts a view of the Eclipse implementing our tool, Fig 1(1) of which shows the editor by which users can write a program. Our proposed tool rewrites the program according to the proper Java programming style, which can be shown as fig. 1(2).

This section explains the outline of the tool using a sample program shown in fig. 2. The program fails to conform to the following guidelines for the programming style of Java:

(a)In the fourth line, the first letter of the class name “jugyou01kadai03” is not a capital letter.

(b)In the fourth line, the class name “jugyou01kadai03” is not an English word.

(c)In the eighth line, the variable “abc” does not have the correct scope.

(d)In the eighth line, the length of the variable name “abc” is too short.

(e)No comments are written in the program.

(f)No indents are inserted in the program.

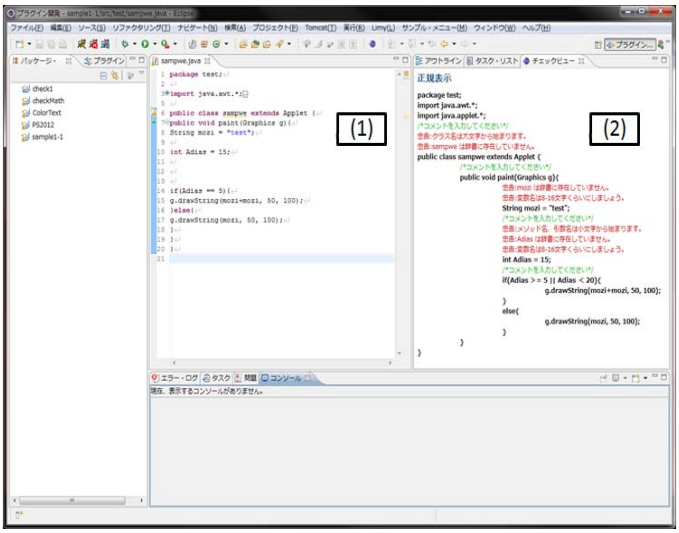

FIG. 1 A VIEW OF ECLIPSE IMPLEMENTED THE TOOL

Fig. 3 indicates the execution result after inputting the program shown in fig. 2 into the tooland shows the source program rewritten in the proper Java programming style as well as displays suggestions if the program has class names, method names and variable names thatis apart from conforming to the programming style. Users of the tool can learn the proper Java programming style by viewing the program as shown in fig. 3 and rewriting and inserting comments into the program in the editor area. Learners cannot edit, such as cut and paste, the program in fig. 3. The procedures are depitedto extract information and implement the functions mentioned above in the following sections.

Extracting Information from the Source Program

The tool extracting information from the source program to implement the functions mentioned above, first performs a lexical analysis and then parses the source program. Finally, the tool extracts the following information:

(1)positions of declared classes, methods, and variables

(2)positions of used variables

(3)positions of instructions, including loops and conditional branches

(4)lines that have an improper indent

(5)lines that have multiple braces

```
1: import java.awt.*;
2: import java.applet.*;
3: import java.awt.event.*;
4: public class jugyou01kadai03 extends Applet
        implements ActionListener {
5: int xPosition = 50;
6: Button moveButton;
7: int numberOfClick = 0;
8: String abc="Executing Func() method";
9: public void init() {
10: moveButton = new Button("move");
11: moveButton.addActionListener(this);
12: add(moveButton);
13: }
14: public void paint(Graphics g) {
15: if (numberOfClick % 5 == 0)
16: xPosition = 50;
17: g.fillRect(xPosition, 50, 50, 50);
18: Func();
19: }
20: public void actionPerformed(ActionEvent e) {
21: numberOfClick++;
22: xPosition += 50;
23: repaint();
24: }
25: public void Func() {
26: System.out.println(abc);
27: }
28: }
```

FIG. 2 EXAMPLE OF A JAVA SOURCE PROGRAM

Recommending the Use of Camelcase

CamelCase is one of Java’s naming guidelines for writing readable and understandable source programs. The following is required for the classes: if the name consists of several words, the first letter of the class name and the first letter of each word are in uppercase. If the first letter of a class name is in lowercase, the tool displays “Use a capital letter for the first letter of the class name,” as shown in fig. 3(a). In addition, the following is required for the methods and variables: if the name consists of several words, the first letter of the methods and variable names is in lowercase, and the first letter of each word is in uppercase. If the first

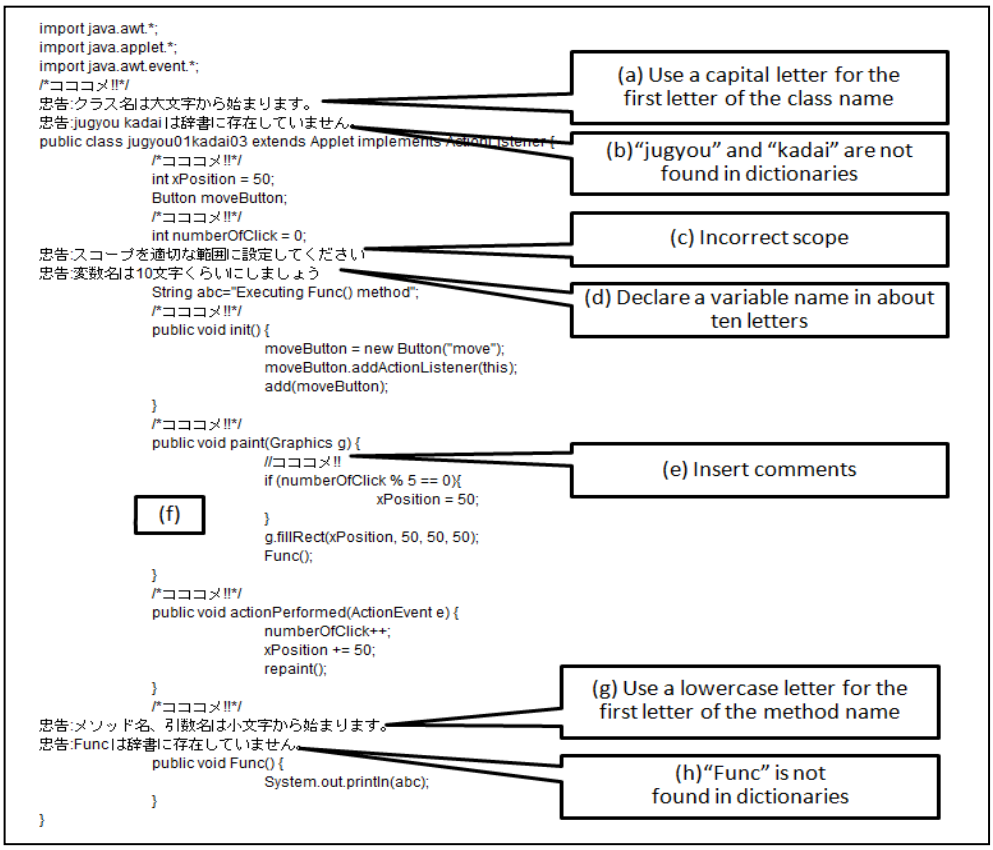

FIG. 3 EXAMPLE OF THE EXECUTION RESULT

letter of a method name is in uppercase, the tool displays "Use a lowercase letter for the first letter of the method name", as shown in fig. 3(g). The tool searches each class, method, and variable name for a uppercase letter and divides the name in front of the uppercase letter. For example, if the name of a class is "ClassName," the tool divides it into "Class" and "Name." Then, the tool searches two dictionaries—an English dictionary and an English abbreviation dictionary—for every word. If the words are not found in the dictionaries, it is likely that the class name does not follow CamelCase and thus, the tool displays "x is not found in dictionaries," as shown in fig. 3(b) and (h), where x is the original class, method, and variable name. The tool displays this message, for example, when the class name is "Classname". We will describe the method of searching dictionaries in the next section.

Recommending the Use of English Words

In companies, computer systems are commonly developed by people from different countries. For this reason, the names of the variables, methods, and classes should be written in English so that everyone can understand them. In this section, we will describe the tool function to recommend the use of English words. If the name does not exist in English dictionaries, the tool displays "x is not found in dictionaries," as shown in fig. 3(b). The class name in "jugyou01kadai03" is shown as an example in fig 2. The two words—"jigyou" and "kadai"—do not exist ("jigyou" and "kadai" in Japanese mean "class" and "subject", respectively). Consequently, the tool displays "jigyou and kadai are not found in dictionaries."

We use Representational State Transfer (REST, 2012) to search English dictionaries. REST is a simple Web-based system with http and XML, and it can implement several functions using the Uniform Resource Locator (URL). Some websites, such as the EAST dictionary webservice (EASR, 2012), use REST to search English dictionaries. We insert a word into the URL to search English dictionaries. Fig. 4(a) and 4(b) show examples of the URL and XML data, respectively, received from the REST site in the case of searching for "camel." The line "<TotalHitCount>1</TotalHitCount>" in fig. 4(b) shows the number of hit counts in the dictionary. In

this case, the dictionary registers one "camel" record. If "<TotalHitCount>" is zero, the tool displays "x is not found in dictionaries."

In a similar manner, the tool can search for abbreviations in the abbreviation dictionary using REST.

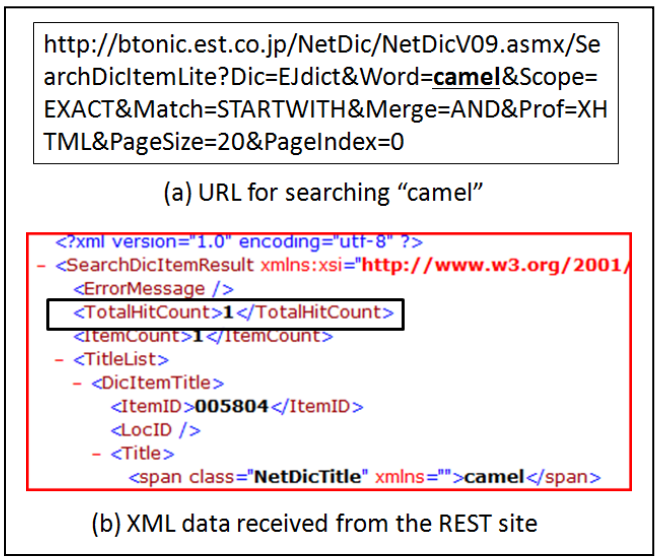

FIG. 4 EXAMPLE OF THE URL AND XML FOR SEARCHING "CAMEL"

Recommending the Correct Scope Level Set up of Variables and the Appropriate Length of Variable Names

Correctly setting the scope level of variables enables easier debugging. Fig. 5 shows an example of the variable scope level. The variables—var1 and var2–are declared at the scope level n, as shown in fig. 5(a). The variable var1 is set at the correct scope level because it is used twice at the scope level n+1, as shown in Fig. 5(b) and (c). The variable var2, however, is used once, as shown in fig. 5(c). Therefore, variable var2 should be declared in area2. The tool can determine the incorrect scope level of variables, as shown in fig. 3(c).

Using the appropriate length of variable names makes source programs more readable and understandable. Gorla et al. stated that a variable name should consist of more than eight, but less than sixteen characters (N. Gorla, 1990). The tool can display "Declare a variable name in about ten letters" as shown in fig. 3(d). However, variable names often used by the loop counter and the array index, such as i, j, and k, are allowable.

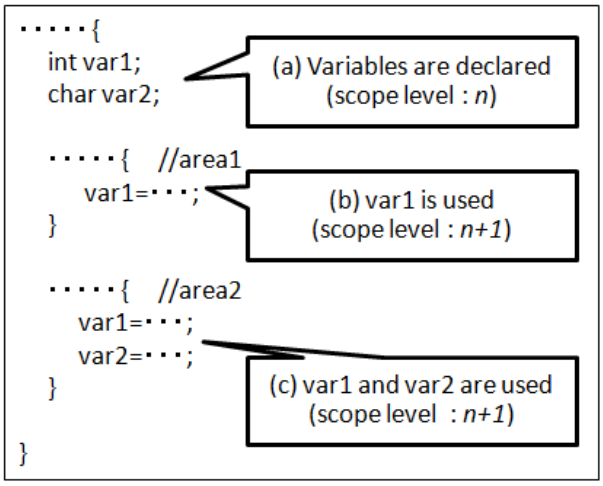

FIG. 5 EXAMPLE OF THE SCOPE LEVEL OF THE VARIABLES

Additional Functions

Our proposed tool has the following additional functions:

Recommendation on Making Comments

As mentioned above, the tool extracts the following positions from a source program: declared classes, methods, variables as well as instructions including loops and conditional branches. The tool displays "inserts comments" above these lines, as shown in fig. 3(e).

With this function, users can practice making comments such as for functions, objectives of classes, behaviors of methods, roles of variables, objectives of loops, and conditional branches.

Exception of File Saving Without Editing

When the user saves the source program, the tool checks whether the user has edited the "insert comments" lines. If not, the tool prohibits saving of the source program.

The Display of a Sample Source Program Following the Programming Style of Java

As mentioned above, the tool extracts lines that have improper indents and multiple braces from the source program and rewrites these lines according to the programming style. The corrected indent lines are shown in fig. 3(f).

Conclusions

In this paper, we proposed an Eclipse plugin tool contributive to learning the proper programming style

of Java has been put forward. The tool has the following functions: recommendation on the use of CamelCase and English words for the names of classes, methods, and variables; recommendation on setting the correct scope level of variables and the appropriate length of variable names; recommendation on making comments in source programs; the showing of sample source programs according to the programming style of Java.

This tool is evaluated by practical use in actual classes.

ACKNOWLEDGMENT

This study was supported in part by the Japan Society for the Promotion of Science, Grant Number (KAKENHI 24501150).

REFERENCES

D. D. Ward and P. H. Jesty, "Guidelines for the use of the C language in critical systems," Motor Industry Research Association, 2004.

EAST dictionaries service http://www.btonic.com/ws/ (Dec. 21, 2012).

Eclipse http://www.eclipse.org/ (Dec. 21, 2012).

K. Suzuki, S. Yokoyama and Y. Miyadera, "Development and evaluation of automatic module division system for programming education," Technical reports of the Institute of Electronics, Information and Communication Engineers ET-2003(697), pp.143-148, 2004. (in Japanese).

M.Arai, "A Tool for Learning the Programming Style of Java", The 2012 International Conference on Software and Intelligent Information, S002, 2012.

M. Oda and T. Kakeshita, "Pitfall detection of C programs using pattern matching," Transactions of Information Processing Society of Japan 35(11), pp.2427-2436, 1994. (in Japanese).

mule http://www.m17n.org/mule/ (Dec. 21, 2012).

N. Gorla, A.C. Benander and B.A. Benander, "Debugging effort estimation using software metrics," IEEE Transactions on Software Engineering SE-16(2), pp.223-231, 1990.

REST http://ja.wikipedia.org/wiki/REST (Dec. 21, 2012).

R. Sekimoto and K. Kaijiri, "A diagnosis system of programming style," Transactions of Japanese Society for Information and Systems in Education 17(1), pp.21-29, 2000. (in Japanese).

Yuki Arakawa graduated from Department of Science and Engineering, Teikyo University, Japan in 2013. He is mainly engaged in developing programming learning tools.

Masayuki Arai is a professor in the Graduate School of Sciences and Engineering at Teikyo University. He received his B.E. degree from Tokyo University of Science in 1981 and Dr. Eng. degree from Utsunomiya University in 1995. His research interests include pattern recognition, natural language processing and information visualization. He is a member of the Information Processing Society of Japan and IEEE.